FIAP ON

DESENVOLVIMENTO DE
APPS – PARTE 1 (ANDROID)

LAYOUTS E COMPONENTES *GRÁFICOS* DE INTERFACE

RAFAEL ACIOLY DE OLIVEIRA

4

LISTA DE FIGURAS

Figura 4.1 – Eixos de uma View.....5
Figura 4.2 – Preview inicial em ConstraintLayout sem a definição de restrições 6
Figura 4.3 – Círculo para definição de dimensões relativas.....7
Figura 4.4 – TextView após definição de dimensões relativas.....7
Figura 4.5 – Configuração de Margem Padrão 8
Figura 4.6 – Configuração de Margem.....8
Figura 4.7 – TextView após configuração de margem 8
Figura 4.8 – Tela com restrições aplicadas em todos os elementos de tela 9
Figura 4.9 – Tela de login com o uso do FrameLayout 15
Figura 4.10 – Tela com login com o uso de TableLayout.....18
Figura 4.11 – Tela com login com o uso de RelativeLayout.....21

LISTA DE CÓDIGOS-FONTE

Código-fonte 4.1 – Código-fonte com ConstraintLayout.....11
Código-fonte 4.2 – FrameLayout com LinearLayout 13
Código-fonte 4.3 – Código-fonte 2 com elementos para tela de login.....14
Código-fonte 4.4 – Tela de Login com TableLayout.....17
Código-fonte 4.5 – Tela de Login com RelativeLayout.....20

SUMÁRIO

4 LAYOUTS E COMPONENTES GRÁFICOS DE INTERFACE 5
4.1 ConstraintLayout 5
4.2 FrameLayout e LinearLayout.....12
4.3 TableLayout.....15
4.4 RelativeLayout.....18
CONCLUSÃO.....22
REFERÊNCIAS.....23

4 LAYOUTS E COMPONENTES GRÁFICOS DE INTERFACE

No Android, existem diferentes tipos de gerenciadores de layout, tais como: *LinearLayout*, *TableLayout*, *RelativeLayout*, *FrameLayout* e *ConstraintLayout*. Alguns podem organizar os componentes na horizontal e na vertical, como também podem organizar os componentes em uma tabela com linhas e colunas.

As próximas seções têm como objetivo apresentar uma tela de autenticação (com caixas de texto, labels e botão), de maneira a entender como essa mesma tela pode ser projetada em cada tipo de layout, apresentando vantagens e desvantagens no uso.

4.1 ConstraintLayout

O *ConstraintLayout* é o layout mais recente oferecido no kit de desenvolvimento para aplicativos Android. A premissa de uso do ConstraintLayout é por meio da definição de restrições, as quais podem ser relativas a outro objeto de tela ou absoluta (definida com medidas exatas). É importante entender, para qualquer layout, os eixos para dimensionamento dos objetos.

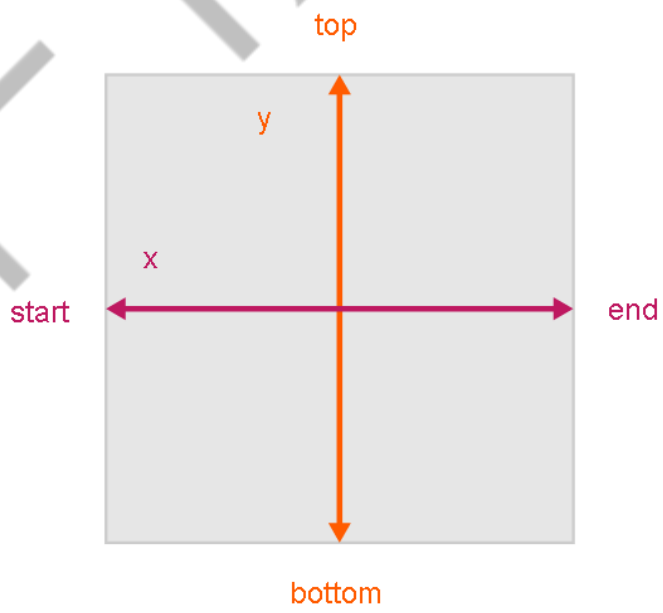

Figura 4.1 – Eixos de uma View
Fonte: Elaborado pelo autor (2019)

A vantagem no uso do ConstraintLayout é a definição prévia do posicionamento dos objetos de tela em modo gráfico (arquivo activity_main.xml, aba *Design*) e, depois, a definição de restrições para o posicionamento de acordo com as necessidades do usuário. Por exemplo, a Figura “Eixos de uma View” apresenta a distribuição de elementos para a tela de autenticação como sugerida anteriormente (sendo a tela em branco a distribuição de elementos proposta e, em azul, com a definição de restrições). Apesar de visualmente o posicionamento dos elementos de tela estarem corretos, ainda falta a definição de restrições.

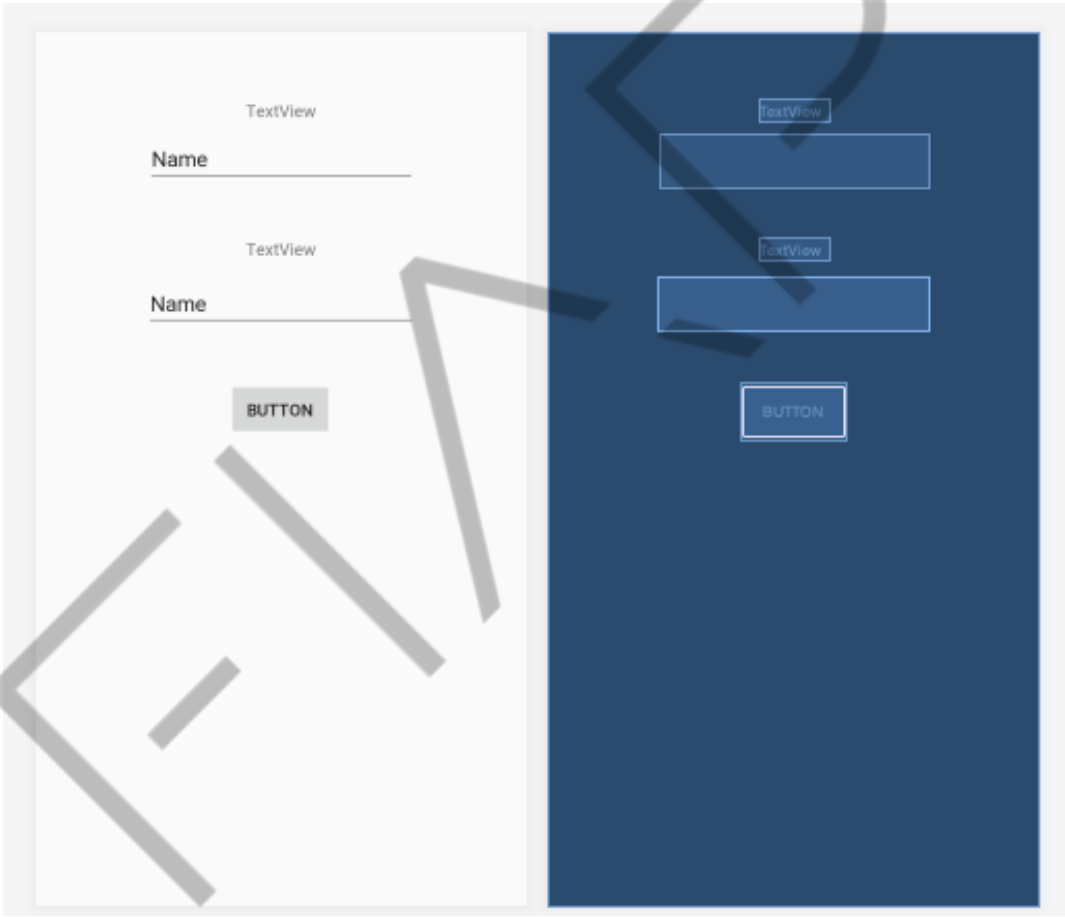

Figura 4.2 – Preview inicial em ConstraintLayout sem a definição de restrições
Fonte: Elaborado pelo autor (2019)

Depois de distribuídos os objetos, devem ser definidas as *constraints* para cada objeto, sendo uma prática semelhante para projetos na plataforma de desenvolvimento em iOS. O que deve ser avaliado na aplicação das restrições são quais dimensões precisam ser fixas e quais outras necessitam ser relativas ao tamanho de tela. As dicas para configurar cada objeto são:

- Ajustar as “forças” para as dimensões vertical e horizontal.

- Depois, caso necessário, definir dimensões absolutas entre os objetos (ou entre o objeto e a borda).

Para ajustar as forças verticais e horizontais de cada objeto, é possível observar marcadores em formato de circunferência nas quatro dimensões (Figura “Círculo para definição de dimensões relativas”). Ao arrastar esse círculo para uma borda ou objeto adjacente, é criada uma “força” que indica que essa dimensão é relativa, apresentada em zigue-zague na Figura “TextView após definição de dimensões relativas”. Isso indica que esse objeto está com restrições verticais e horizontais relativas. O segundo passo é definir dimensões absolutas em eixos nos quais é necessário manter um espaçamento fixo.

Figura 4.3 – Círculo para definição de dimensões relativas
Fonte: Elaborado pelo autor (2019)

Figura 4.4 – TextView após definição de dimensões relativas
Fonte: Elaborado pelo autor (2019)

Observe na Figura “TextView após definição de dimensões relativas” que, ao serem definidas as restrições relativas, existe uma margem absoluta previamente definida de 8dp. Essa margem padrão pode ser configurada no ambiente de design do Android Studio, como mostra a Figura “Configuração de Margem Padrão”.

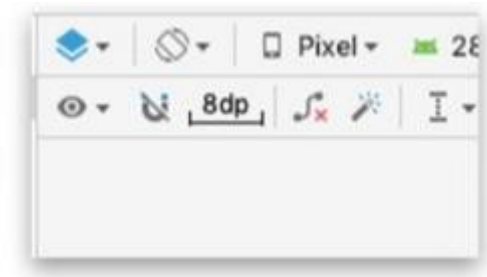

Figura 4.5 – Configuração de Margem Padrão
Fonte: Elaborado pelo autor (2019)

Quando um objeto é selecionado, também são apresentadas as propriedades de Layout na área Attributes, na tela de Design (Figura “Configuração de Margem”), na qual, é possível fazer esse ajuste absoluto. Por exemplo, a dimensão “top” será ajustada em 50dp em relação à borda superior, ou seja, para qualquer dimensão de tela, o espaçamento entre o topo desse objeto e a margem superior sempre será o mesmo. A Figura “TextView após configuração de margem” mostra o TextView, após a configuração de margem.

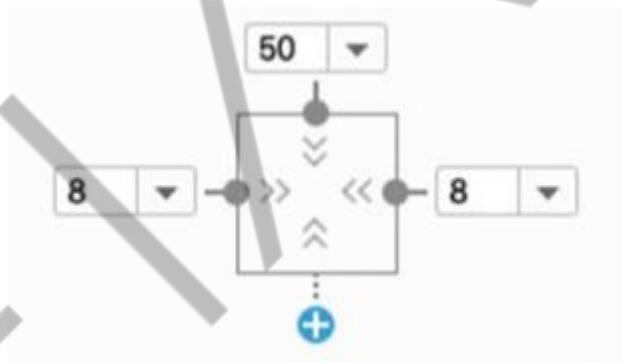

Figura 4.6 – Configuração de Margem
Fonte: Elaborado pelo autor (2019)

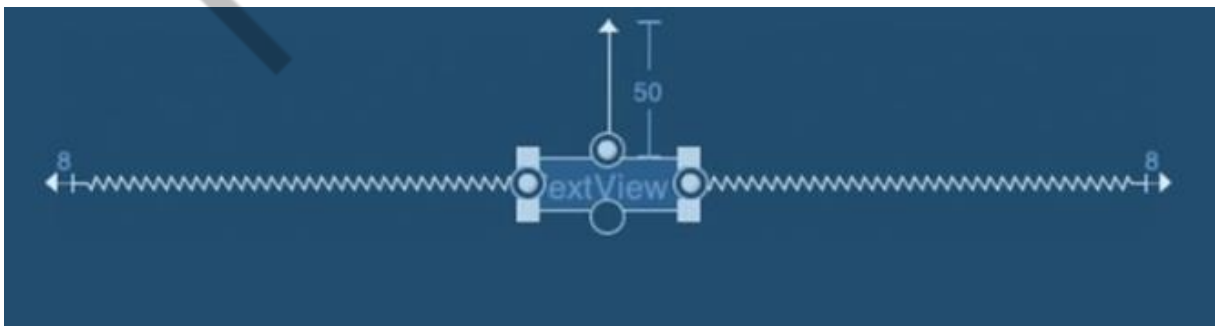

Figura 4.7 – TextView após configuração de margem
Fonte: Elaborado pelo autor (2019)

Com isso, todos os elementos de tela podem ter suas restrições e margens configuradas seguindo as mesmas premissas. A Figura “Tela com restrições aplicadas

em todos os elementos de tela" apresenta todos os objetos de tela configurados. O Código-fonte "Código-fonte com ConstraintLayout" apresenta o XML gerado. Observe que existem *tags* que determinam a restrição (constraint) e outros que determinam a margem. Ajustes podem ser realizados em modo texto (tal como a propriedade text dos TextView e Button), entretanto, o ConstraintLayout possui uma maneira de ajuste menos programável e mais configurável em ambiente gráfico, seguindo uma linha de configuração de layout semelhante ao Apple Xcode.

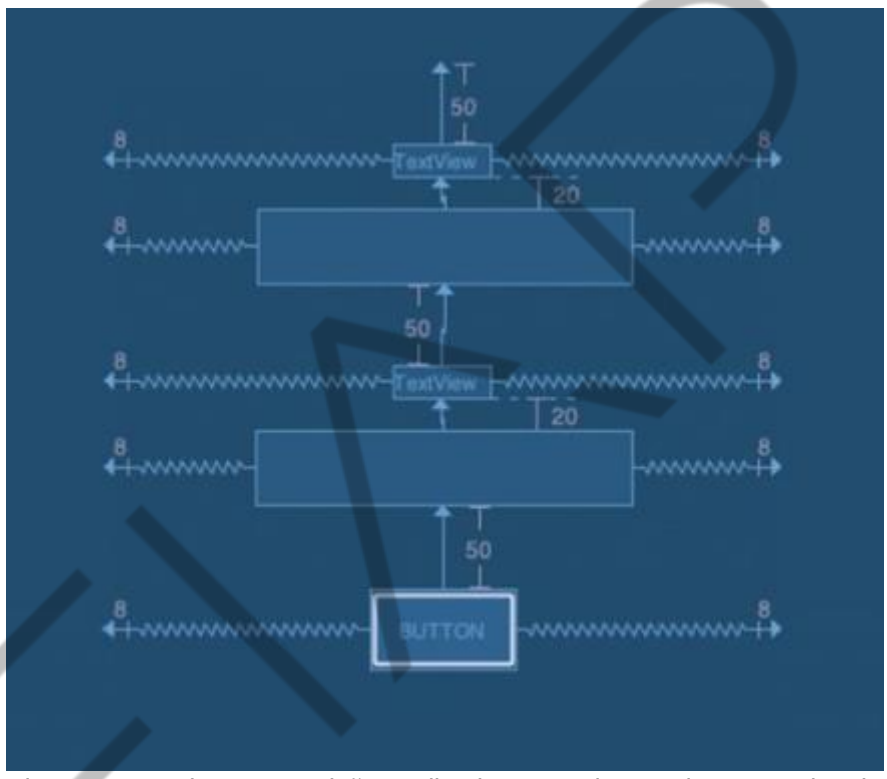

Figura 4.8 – Tela com restrições aplicadas em todos os elementos de tela
Fonte: Elaborado pelo autor (2019)

```
      <?xml          version="1.0"          encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
    <TextView
        android:text="Usuário"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginEnd="8dp"
app:layout_constraintStart_toStartOf="parent"
        android:layout_marginStart="8dp"
android:layout_marginTop="50dp"
        app:layout_constraintTop_toTopOf="parent"/>
    <EditText
        android:layout_width="226dp"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/editText"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
android:layout_marginStart="8dp"
        app:layout_constraintStart_toStartOf="parent"
android:layout_marginTop="20dp"
        app:layout_constraintTop_toBottomOf="@+id/textView"
app:layout_constraintHorizontal_bias="0.502"/>
    <TextView
        android:text="Senha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginEnd="8dp"
app:layout_constraintStart_toStartOf="parent"
        android:layout_marginStart="8dp"
android:layout_marginTop="50dp"
```

```
        app:layout_constraintTop_toBottomOf="@+id/editText"/>
    <EditText
        android:layout_width="227dp"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/editText2"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
android:layout_marginStart="8dp"
        app:layout_constraintStart_toStartOf="parent"
android:layout_marginTop="20dp"

app:layout_constraintTop_toBottomOf="@+id/textView2"/>
    <Button
        android:text="OK"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginEnd="8dp"
app:layout_constraintStart_toStartOf="parent"
        android:layout_marginStart="8dp"
android:layout_marginTop="50dp"

app:layout_constraintTop_toBottomOf="@+id/editText2"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Código-fonte 4.1 – Código-fonte com ConstraintLayout.
Fonte: Elaborado pelo autor (2020)

4.2 FrameLayout e LinearLayout

Pode-se dizer que o uso do FrameLayout e LinearLayout remete ao desenvolvimento de layouts responsivos para web devido ao fato de ser um meio com um viés mais programático e trabalhar com os objetos de tela baseado em empilhamento. Diferente do ConstraintLayout, o FrameLayout permite definir um perfil de ocupação de espaço do objeto e suas margens. Perfis de posicionamento podem ser utilizados por meio da tag *gravity*.

Primeiramente, deve-se definir a tag raiz para FrameLayout. Depois, no exemplo em questão, será utilizado um LinearLayout dentro de um FrameLayout para empilhamento vertical dos objetos (definido pela tag orientation). Também é importante observar as tags layout_width e layout_height: elas possuem as definições-chave para o dimensionamento do objeto de tela.

Caso a dimensão seja configurada para *match_parent*, o objeto de tela ocupará o maior espaço possível na dimensão desejada. Já quando configurado com *wrap_content*, ocupará o mínimo espaço possível na dimensão desejada. O Código-fonte “FrameLayout com LinearLayout” apresenta uma primeira etapa de construção da tela de login.

```
     <?xml          version="1.0"          encoding="utf-8"?>
<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <LinearLayout       android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:orientation="vertical">
```

```
        </LinearLayout>

</FrameLayout>
```

Código-fonte 4.2 – FrameLayout com LinearLayout
Fonte: Elaborado pelo autor (2019)

Com isso, todos os elementos filhos de LinearLayout serão empilhados verticalmente. O próximo passo é inserir dois TextView, dois EditText e o Button, definindo as propriedades necessárias. O Código-fonte “Código-fonte 2 com elementos para tela de login” mostra o código XML com os widgets inseridos. É possível observar no LinearLayout a adição da tag *padding*, que permite definir um espaçamento das arestas do objeto para o centro e, assim, “descolar” o LinearLayout em 50dp em relação à borda.

Outra observação é a configuração de margem (já utilizada na seção anterior) e também a propriedade *gravity*, que permite o alinhamento do objeto em relação ao pai. Já com essas configurações, é possível observar a previsão de layout assim como manter a tela responsiva (Figura “Tela de login com o uso do FrameLayout”).

```
    <?xml version="1.0" encoding="utf-8"?>
<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <LinearLayout       android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:orientation="vertical"
                    android:padding="50dp">
```

```
            <TextView   android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:text="Usuário"
                        android:textAlignment="center"/>

            <EditText   android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="30dp"/>


            <TextView   android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:text="Senha"
                        android:textAlignment="center"
                        android:layout_marginTop="30dp"/>

            <EditText   android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="30dp"
                        android:inputType="textPassword"/>

            <Button     android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="OK"
                       android:layout_gravity="center"
                       android:layout_marginTop="50dp"/>

        </LinearLayout>

</FrameLayout>
```

Código-fonte 4.3 – Código-fonte 2 com elementos para tela de login
Fonte: Elaborado pelo autor (2019)

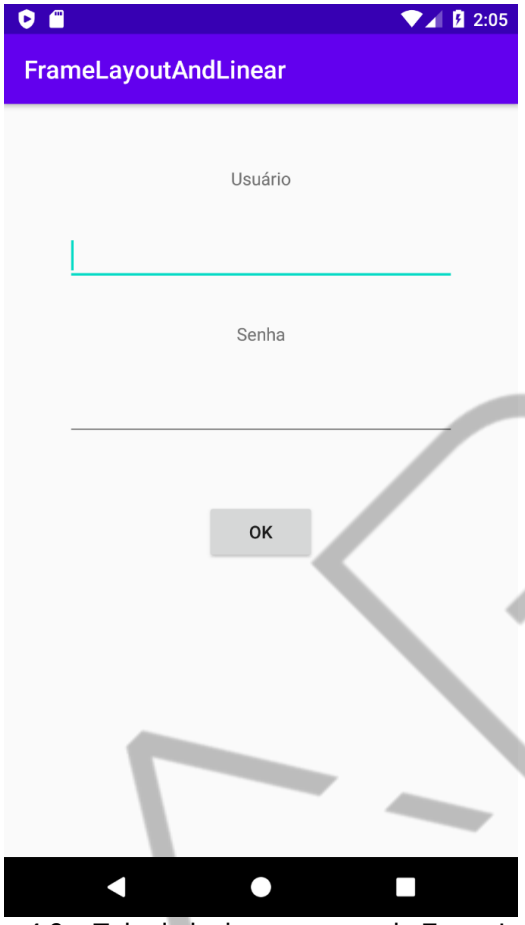

Figura 4.9 – Tela de login com o uso do FrameLayout
Fonte: Elaborado pelo autor (2020)

Dica: Caso seja necessário que a tela tenha barra de rolagem, basta trocar o FrameLayout por ScrollView no objeto raiz. Na ScrollView também é necessário definir a orientação da barra de rolagem (vertical ou horizontal).

4.3 TableLayout

O TableLayout é um LinearLayout o qual permite o empilhamento de linhas contendo colunas, elementos esses chamados de TableRow. Desse modo, seguindo o exemplo da tela de login, o TextView e o EditText de usuário e senha serão dimensionados em duas TableRow. Por sua vez, o botão será empilhado logo após as TableRow, se comportando como uma linha vertical do LinearLayout. O Código-fonte “Tela de Login com TableLayout” apresenta um exemplo do uso do TableLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="50dp"
        android:stretchColumns="1"
        tools:context=".MainActivity">

        <TableRow>
                <TextView   android:layout_width="match_parent"
                            android:layout_height="wrap_content"
                            android:text="Usuário"
                            android:layout_marginRight="10dp"/>

                <EditText   android:layout_width="match_parent"

android:layout_height="wrap_content"/>

        </TableRow>

        <View               android:layout_width="match_parent"
             android:layout_height="50dp"/>

        <TableRow>
                <TextView   android:layout_width="match_parent"
                            android:layout_height="wrap_content"
                            android:text="Senha"
                            android:layout_marginRight="10dp"/>

                <EditText   android:layout_width="match_parent"

android:layout_height="wrap_content"/>
```

```
        </TableRow>

        <Button              android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="OK"
                android:layout_gravity="center"
                android:layout_marginTop="50dp"/>


</TableLayout>
```

Código-fonte 4.4 – Tela de Login com TableLayout
Fonte: Elaborado pelo autor (2019)

É possível observar, no Código-fonte “Tela de Login com TableLayout”, o uso de uma View entre as TableRow, de forma a gerar um espaço entre as linhas. A propriedade *stretchColumns* também identifica as colunas nas quais deseja-se ocupar o máximo espaço com identificação da esquerda para a direta, a partir do número 0. A Figura “Tela de login com o uso do FrameLayout” apresenta uma prévia da distribuição dos objetos.

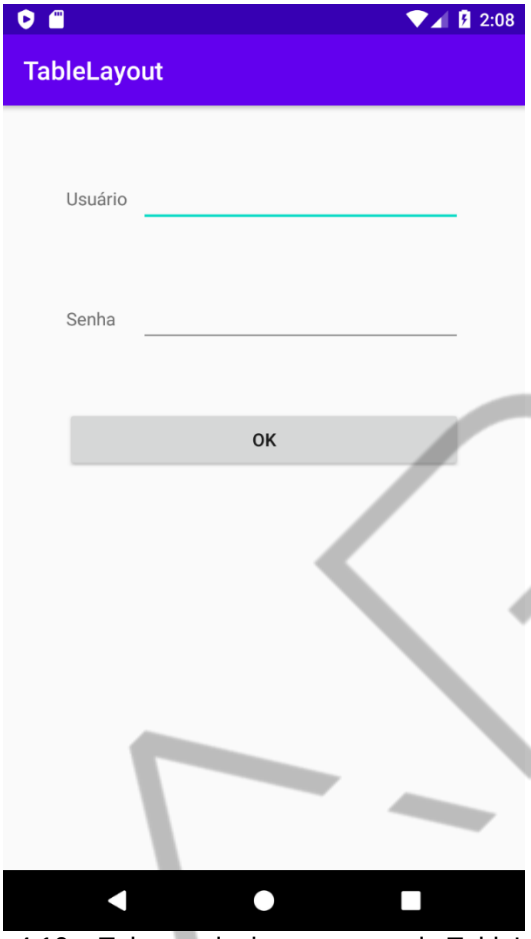

Figura 4.10 – Tela com login com o uso de TableLayout
Fonte: Elaborado pelo autor (2020)

4.4 RelativeLayout

O *RelativeLayout* é considerado o layout de maior dificuldade de implementação, considerando o fato de que um componente de tela depende exclusivamente do posicionamento do anterior. Seguindo o exemplo da tela de login, serão dimensionados dois TextView, dois EditText e um Button, buscando o posicionamento semelhante a exemplos anteriores. O Código-fonte “Tela de Login com RelativeLayout” apresenta um código de exemplo.

Dica: É importante reforçar que — para o uso do RelativeLayout — todos os elementos de tela precisam ter um ID definido.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView           android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="Usuário"
                  android:layout_marginTop="50dp"
                  android:textAlignment="center"
                  android:id="@+id/lbUsuario"/>

        <EditText           android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="30dp"
                  android:layout_marginLeft="50dp"
                  android:layout_marginRight="50dp"
                  android:id="@+id/txtUsuario"
                  android:layout_below="@id/lbUsuario"/>

        <TextView           android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="Senha"
                  android:layout_marginTop="50dp"
                  android:textAlignment="center"
                  android:id="@+id/lbSenha"
                  android:layout_below="@id/txtUsuario"/>

        <EditText           android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="30dp"
                  android:layout_marginLeft="50dp"
```

```
                android:layout_marginRight="50dp"
                android:id="@+id/txtSenha"
                android:layout_below="@id/lbSenha"/>

        <Button             android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="OK"
                android:layout_gravity="center"
                android:id="@+id/btOk"
                android:layout_marginTop="50dp"
                android:layout_below="@+id/txtSenha"
                android:layout_centerInParent="true"/>

</RelativeLayout>
```

Código-fonte 4.5 – Tela de Login com RelativeLayout
Fonte: Elaborado pelo autor (2019)

A complexidade de uso, nesse caso, está em entender o melhor posicionamento relativo para cada elemento. Foi, portanto, utilizado o *layout_below* (que referencia o objeto que está abaixo de outro) e a propriedade *centerInParent*, em que pode ser configurado verdadeiro ou falso para os casos nos quais se deseja centralizar um elemento em função do anterior (o botão em relação à caixa de texto de senha). A Figura “Tela de login com o uso do RelativeLayout” mostra uma prévia da tela de login com o uso do RelativeLayout.

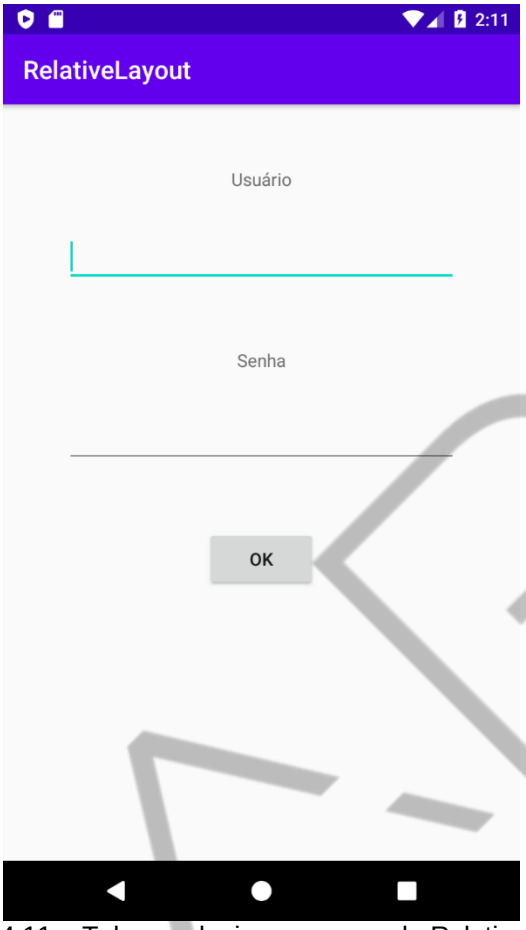

Figura 4.11 – Tela com login com o uso de RelativeLayout
Fonte: Elaborado pelo autor (2020)

CONCLUSÃO

O ambiente de desenvolvimento para aplicativos em Android possui diferentes maneiras de utilizar recursos de configurações de layout, podendo o desenvolvedor escolher o melhor que for aplicado ao problema e de melhor domínio. Nos exemplos desenvolvidos, todos podem ter continuidade com a geração de eventos e transições de tela ou serviço desejados (conhecimentos fornecidos nos capítulos anteriores).

REFERÊNCIAS

ANDROID. **Guia do Usuário**. 2020. Disponível em: <https://developer.android.com/?hl=pt-br>Acesso em: 06 out. 2020.

ANDROID. **Reference Documentation** 2020. Disponível em: <https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout>. Acesso em: 06 out. 2020.

LECHETA, R. R. **Android Essencial com Kotlin**. São Paulo: Novatec, 2017.